Atmel AVR4029: Atmel Software Framework User Guide

Features

- **Getting Started**
- **Access to ASF examples and applications**
- **Add ASF modules to a project**
- **Atmel® Studio 6**
- **IAR™**
- **GNU makefile / GCC**

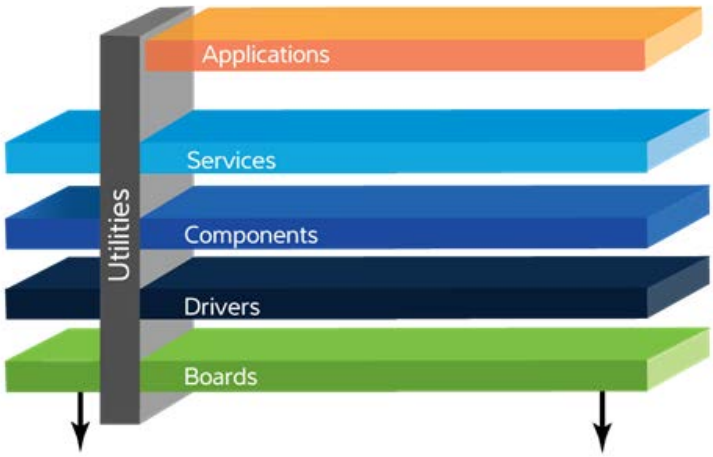
1 Introduction

The Atmel software framework (ASF, www.atmel.com/asf) provides software drivers and libraries to build applications for Atmel megaAVR®, AVR XMEGA®, AVR UC3 and SAM devices. It has been designed to help develop and glue together the different components of a software design. It can easily integrate into an operating system (OS) or run as a stand-alone product.

In this application note developers can read about how to get started with Atmel Studio 6, IAR and GNU GCC makefile:

- How to install the ASF
- How to start ASF reference applications
- How to get ASF documentation

It is recommended but not required to read the ASF reference manual for advanced knowledge on the ASF architecture to read this getting started document.



Atmel Microcontrollers

Application Note

Rev. 8431C-AVR-03/2013

2 ASF Overview

The following is a basic overview of the ASF. For more details, consider reading the ASF Reference manual.

2.1 ASF Layers

The ASF is basically organized in layers for each Atmel family of devices (see figure in Introduction).

2.2 ASF Items

There are mainly three kinds of items in the ASF:

- ASF Module: a software entity that does not have an *int main(void)* function (e.g. a USART software driver, a USB stack). ASF modules are provided by the Drivers, Components and Services layers. An ASF module may be statically and/or dynamically configurable.
- ASF Application (aka ASF Example): a software entity in the ASF that can be built into an executable. This can be: a demo application, or an example of usage of an ASF module. ASF applications are provided by the Drivers, Components, Services and Applications layers. An ASF application used as an example of usage of one ASF module is called an ASF Example. An ASF application may be statically configurable.
- ASF Board: abstraction layer for each Atmel kit supported in the ASF. Usually associated with one device. The “special” board named “User board” is an empty board abstraction layer used to create a user-specific/custom board.

2.3 ASF Ports

There are four ports of the ASF, one for each Atmel family of devices: megaAVR, AVR XMEGA, AVR UC3 and SAM.

The common top folder gathers modules and applications shared amongst several architectures: in some cases the implementations are shared while for some modules only the API is common. For the latter, the common folder thus also contains ports for each Atmel family of devices.

The thirdparty top folder contains modules and applications with licenses other than the Atmel ASF license. As for the common folder, thirdparty is not dedicated to one family but contains instead ports to for each Atmel family of devices.

2.4 ASF Directory Structure

There is a top level folder for each Atmel family of devices as well as the common and thirdparty folders.
The organization beneath each of these folders is following the ASF layers split (applications, services, components, drivers, boards).
Note that the folder of an ASF module always contains at least an example of usage which is an ASF item of type ASF application.

2.5 ASF Releases Formats

Each ASF release is fully documented and available online at asf.atmel.com.

Each ASF version is released in two ways:

- As an extension to Atmel Studio 6,
- Stand-alone zip package of the whole ASF directory tree with, for each ASF application, a make project and an IAR project.

Note that Atmel Studio 6 has dedicated menus and actions for the ASF (which are presented later in the document).

3 Getting started with ASF and Atmel Studio 6

Atmel Studio 6 is the integrated development environment (IDE) for developing and debugging Atmel ARM® Cortex™-M and Atmel AVR® microcontroller (MCU) based applications.
Content and features of Atmel Studio 6:

- Solutions and projects creation,
- a powerful editor with visual assist and plug-ins support to enhance the IDE,
- a GNU C/C++ Compiler for generating target executables,
- the ASF and ASF-oriented wizards,
- Programming & Debugging mode supporting all Atmel debuggers, programmers and the AVR simulator. In debugging mode, the IDE presents MCU status in nicely formatted views, giving fast access to critical system parameters.
- Atmel Gallery

3.1 Installation - Requirements

3.1.1 Software Tools

1. Download Atmel Studio 6 on http://www.atmel.com/atmelstudio. The Atmel ASF is included in Studio 6 and does not require a separate download.
2. Install.
3. Launch Atmel Studio: refer to embedded Atmel Studio user guide and getting started in the welcome screen to get started with Atmel Studio.

Further ASF updates: get ASF updates from the Extension Manager.

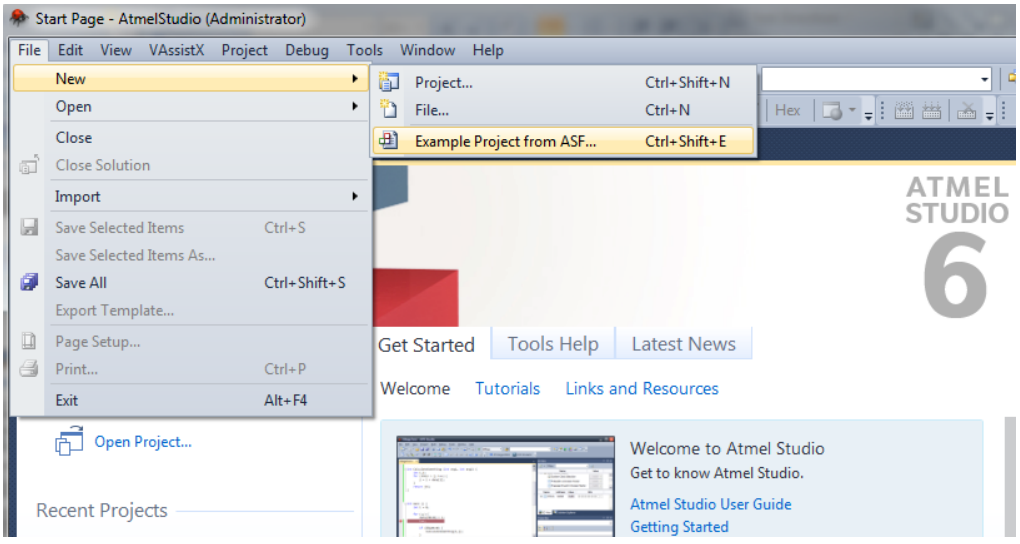
3.1.2 Hardware Tools

On top of the AVR simulator provided with Atmel Studio 6, Atmel Studio 6 connects to all hardware kits, programmers and debuggers supporting all existing Atmel devices.

3.2 Start ASF examples

1. In the file menu, select New -> Example Project from ASF.

Figure 3-1. New example project from ASF.



It is possible to sort the ASF examples by kits (for example, Atmel EVK1100, Atmel AVR Xplained…), by technology keywords (for example, USB, IO, Interrupt…) or by category (drivers, components, services, applications). The example documentation can be viewed by clicking the help icon in the right panel of the New Example Project window. Find the project "USART Example – XMEGA-A1 Xplained" and click OK to create it.

Figure 3-2. Selecting a new ASF example for a kit.

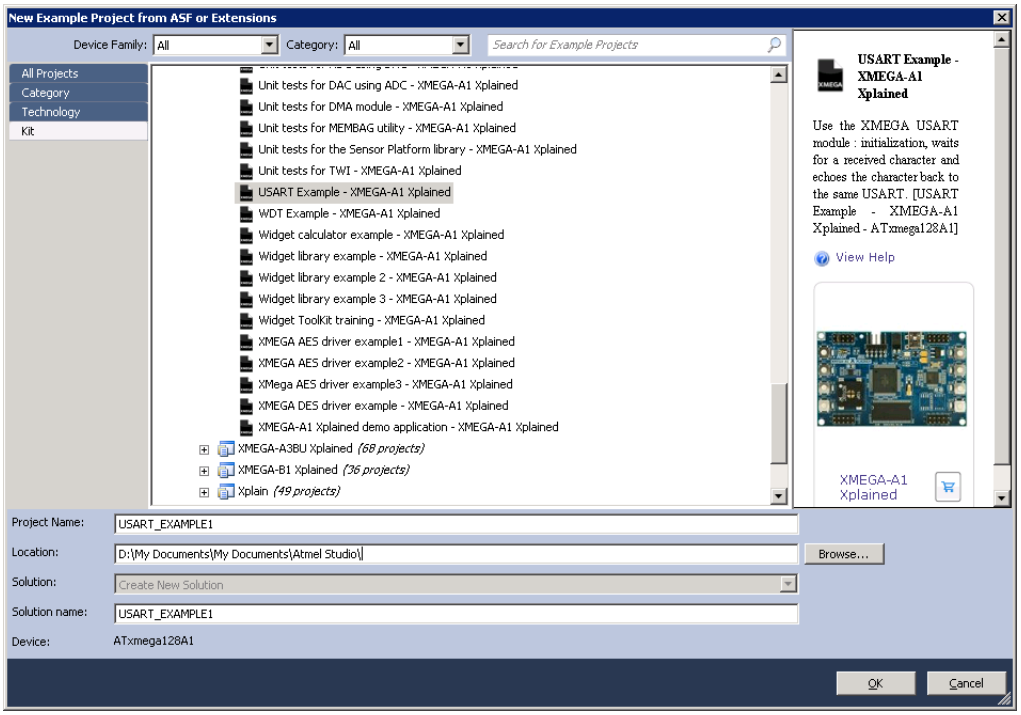
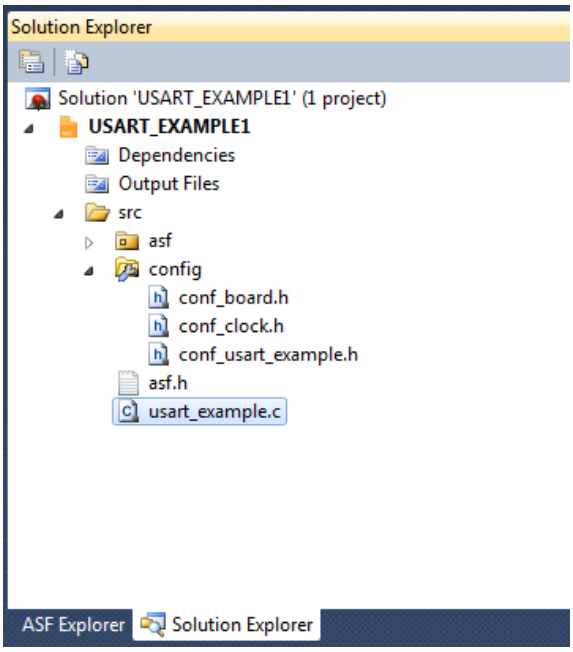


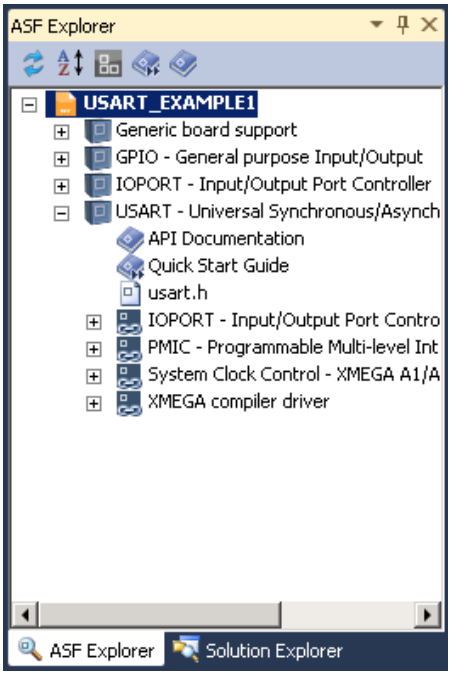
Figure 3-3. ASF examples solution explorer view.



2. It is now possible to compile, load and debug the project. The Atmel ASF files are imported into a new Atmel Studio project in the src folder. Example files are located in the src/ folder. Dependency files are located in the src/asf/ folder. Connect a debugger to the PC and run the ASF example.

3. The ASF Explorer is a logical view of ASF code, with easy access to API and documentation.

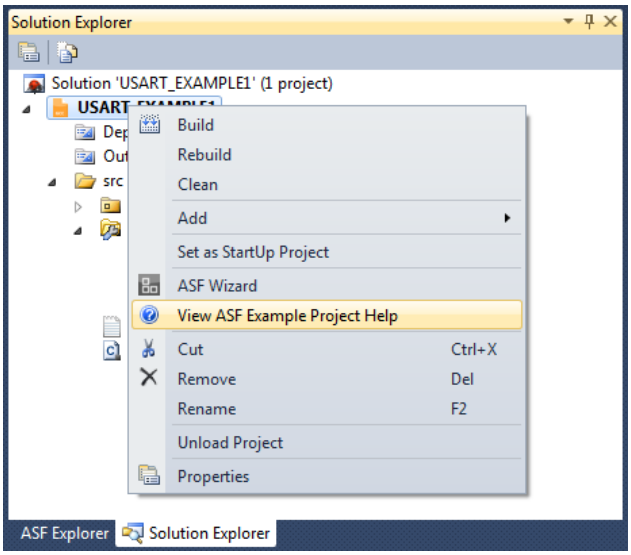
Figure 3-4. ASF Explorer view.



3.3 Get ASF examples documentation

- Right click on the project name
- Select “View ASF Project Examples help”

Figure 3-5. ASF examples documentation.

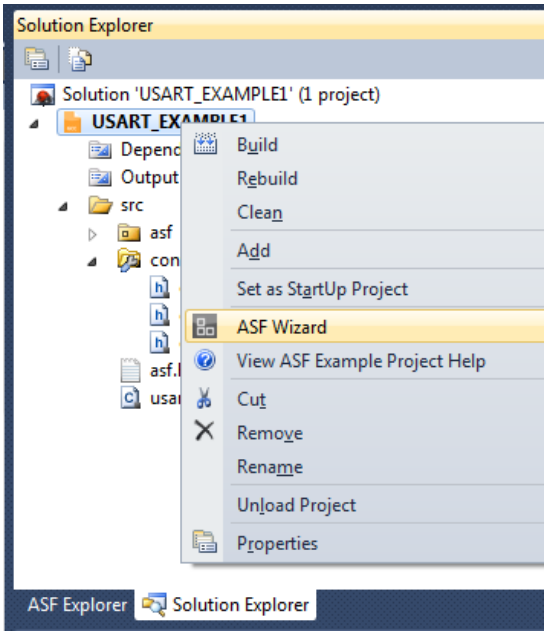


3.4 Add ASF modules to an existing project

It is possible to add/remove ASF modules to any project.

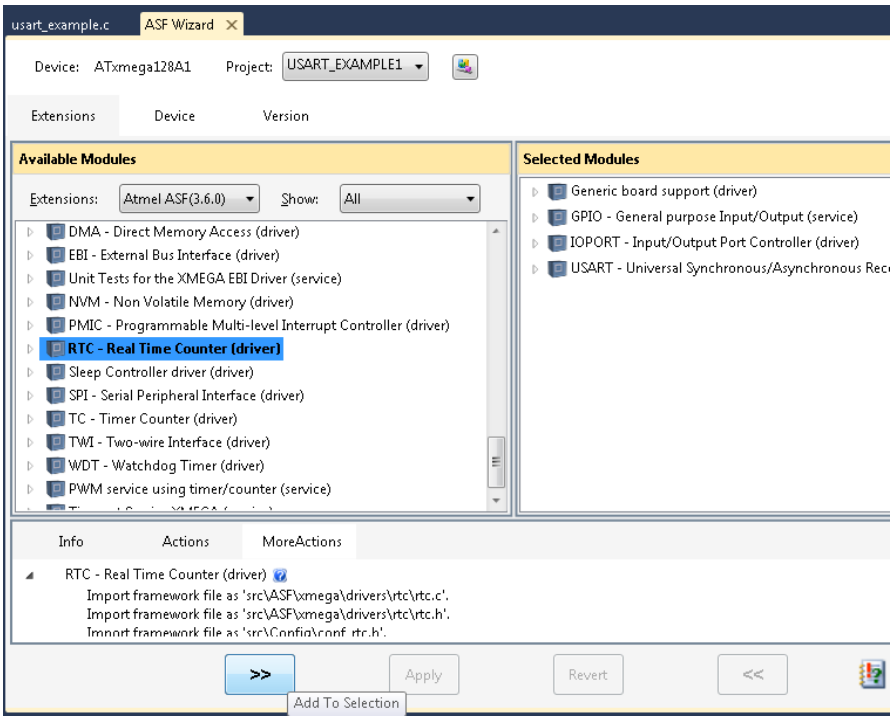
1. Right click on the project or project menu, and then click on “ASF Wizard”.

Figure 3-6. Adding Atmel ASF modules to a project.



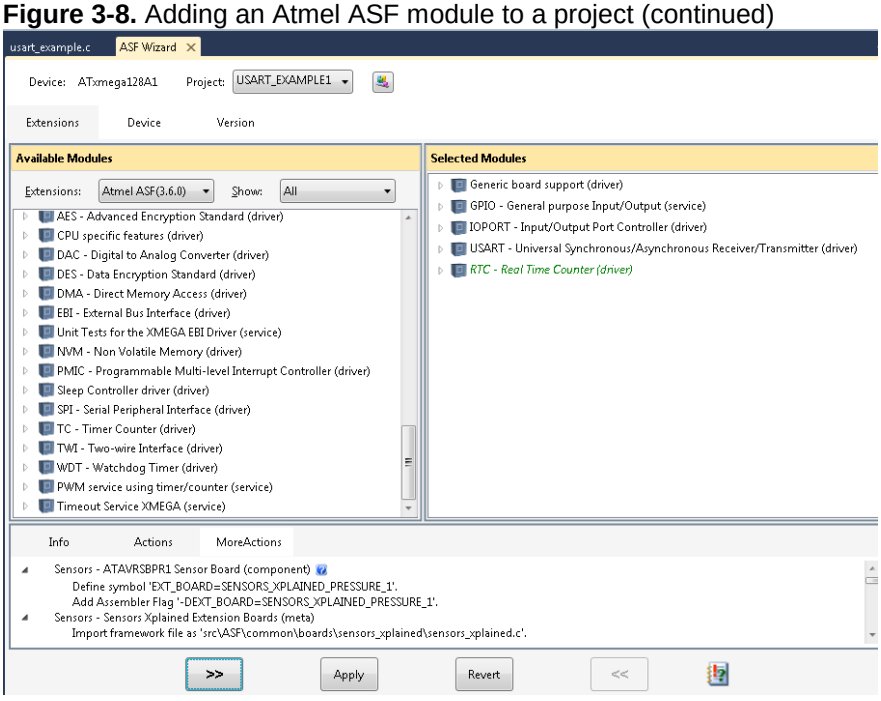
2. Select one or several ASF modules, and then click on “Add to selection” (displayed as “>>”). In this example, the RTC driver is selected.

Figure 3-7. Adding an Atmel ASF module to a project.



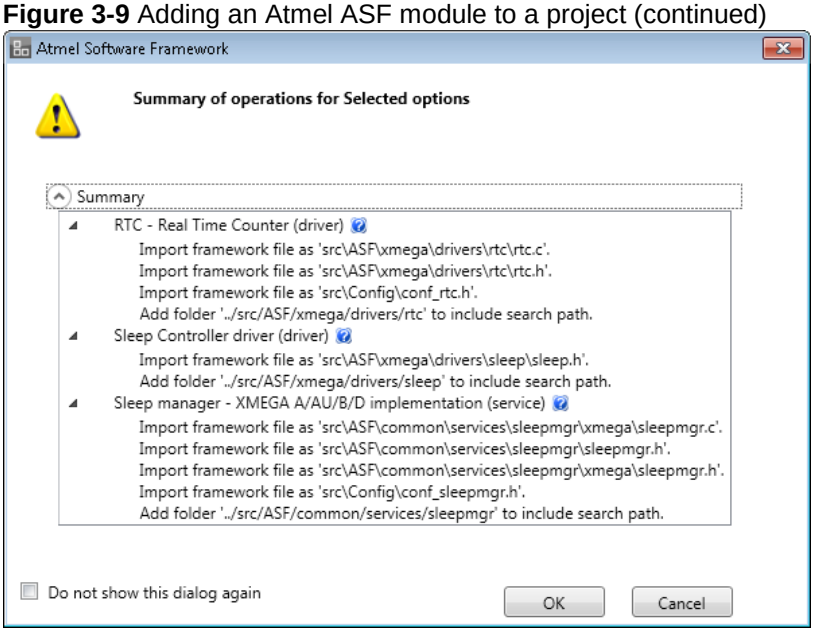
3. The RTC driver is then added to the “Selected Modules” list in green and in italic. This means it is not yet fully added and requires confirmation. Click on the “Apply” button for that.

Figure 3-8. Adding an Atmel ASF module to a project (continued)



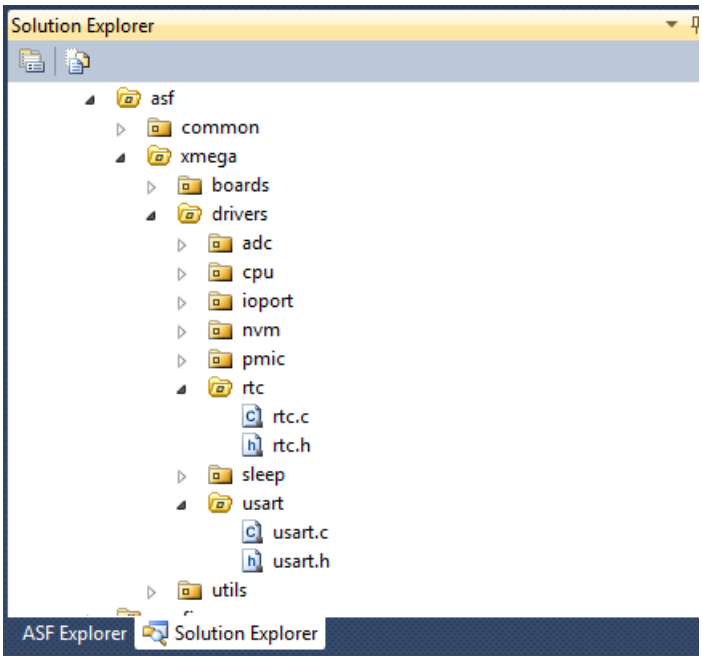
4. After clicking on “Apply”, a pop-up window appears summarizing the addition/deletion operations that will be performed on the current project. Note that non-selected modules may appear due to dependencies (for example the RTC driver uses the Sleep manager which relies on the Sleep Controller driver (see screenshot here below)).

Figure 3-9 Adding an Atmel ASF module to a project (continued)



5. Finally, click on “OK”. For example, the RTC – Real Time Counter driver rtc.c and rtc.h files are added in the src/asf/xmega/drivers/rtc folder.

Figure 3-10 Solution explorer of an ASF example.



3.5 Start a new project with a template: User Application Template

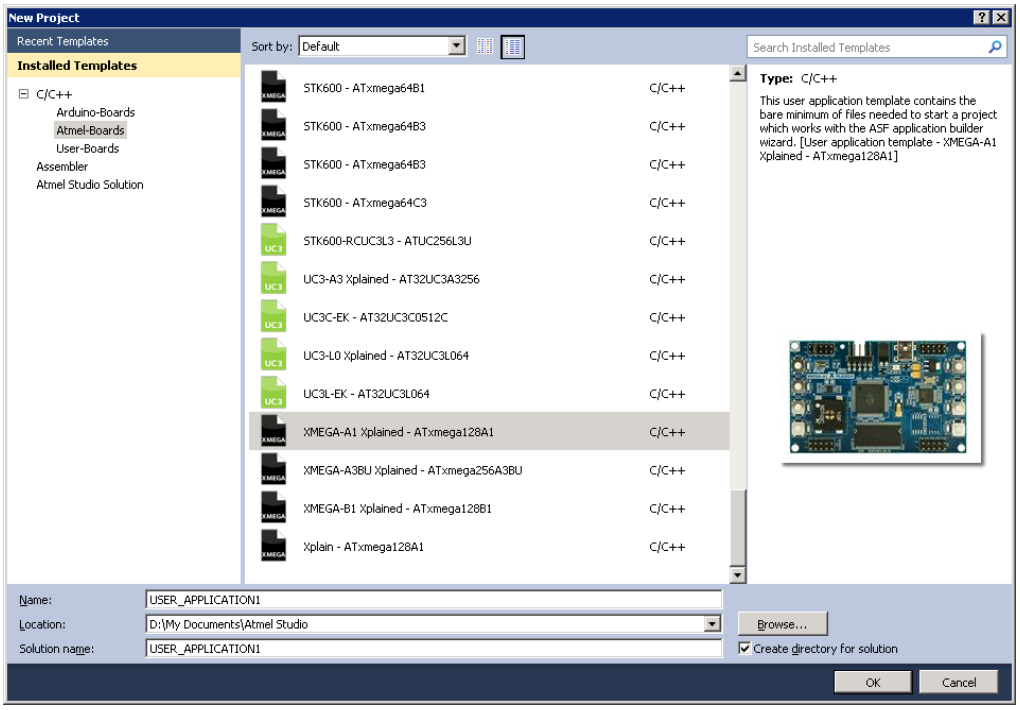
User application templates are ready to use projects with:

- A main.c file and a main() function
- A basic set of ASF drivers (GPIO, interrupt) located in src/asf/ folder for the selected Atmel MCU
- An ASF board definition file. For example, for the Atmel XMEGA-A1 Xplained kit, the src/asf/xmega/boards/xmega_a1_xplained/xmega_a1_xplained.h file is added.

To access User Application Template:

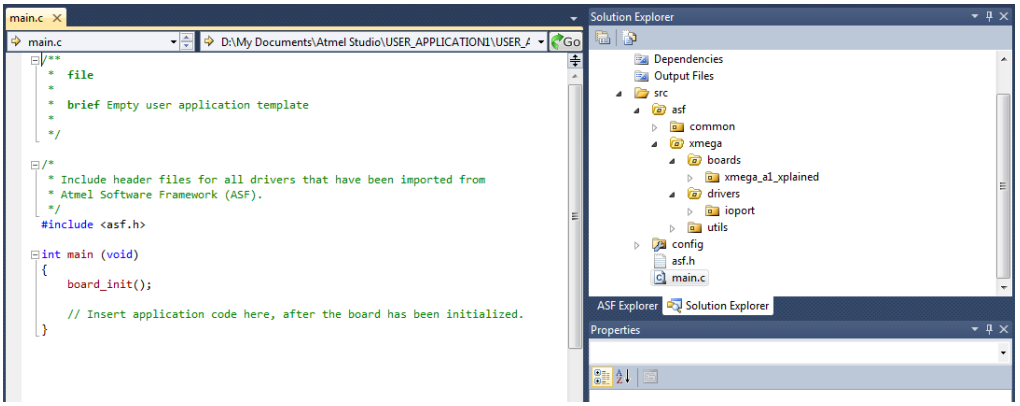
1. Click on File->New -> Project.
2. In C/C++-> Atmel Boards menu, select the User Application Template for the wanted Atmel board.

Figure 3-11. New user template project for an Atmel board.



3. Click OK.

Figure 3-12. New user template project for the Atmel XMEGA-A1 Xplained.

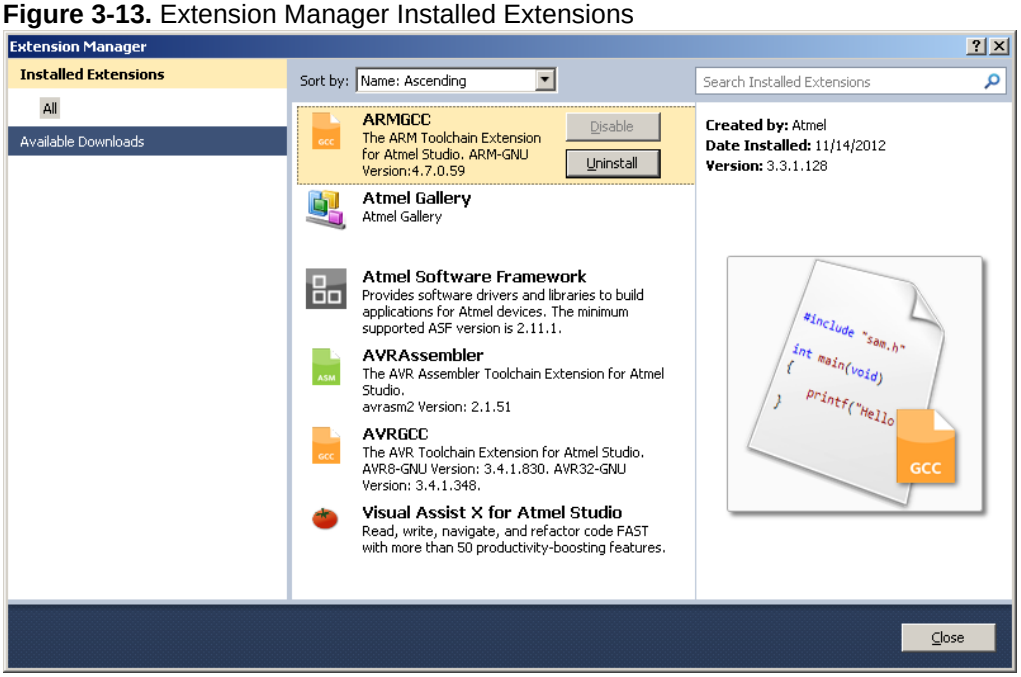


3.6 Using the Atmel Gallery

The Atmel Gallery contains extensions to the Atmel Studio platform in the form of development tools or embedded software.

The Atmel Gallery is accessible through the Extension Manager. The Extension Manager is used to add, remove, enable, and disable Atmel Studio extensions. To open the Extension Manager, on the Tools menu, click Extension Manager.

Figure 3-13. Extension Manager Installed Extensions



For more information see the Atmel Studio Help (ctrl-F1), section “Extending Atmel Studio”. For installing new extensions in Atmel Studio, select the step by step section “Installing new extensions in Atmel Studio”. Check also the “XDK – User Guide” document available on atmel.com.

3.7 Video

Getting started with a video of Atmel Studio and ASF, follow Atmel videos on http://www.youtube.com/user/AtmelCorporation.

4 ASF-based Development Flow in Atmel Studio

Given the numerous available ASF Modules, Applications and Boards provided by the ASF and the ASF-based wizards provided in Atmel Studio, there is more than one way to develop an application based on the ASF. The purpose of this section is to guide the user on how to use the ASF in Atmel Studio along a typical development flow.

We will first identify a list of development scenarios then we'll see how each can be done using the ASF in Atmel Studio.

4.1 Development Types: Overview

There are mostly, if not only, two main kind of development with the ASF:

- on an Atmel board,
- on a custom board (aka user board).

4.1.1 Development on an Atmel Board

There are mainly four scenarios:

- Run (and eventually modify) an existing ASF Example on an Atmel Referenced Board
- Develop a New ASF-based Application on an Atmel Referenced Board
- Using the ASF, port a non-ASF existing application on an Atmel Referenced Board
- Develop a non-ASF-based Application on an Atmel Referenced Board: out of scope.

4.1.2 Development on a Custom Board (aka User Board)

We will consider only the most relevant scenario: develop a Custom Application on a Custom Board. There are mainly three approaches to do that:

- Modify an existing ASF example that is the closest to the target application, port it to the custom board, then modify the application as needed
- Develop a New ASF-based Application to a Custom Board
- Using the ASF, port a non-ASF existing application to a Custom Board
- Develop a non-ASF-based Application on a custom board: this is out of this section's scope

4.1.3 Release on the Atmel Gallery

Since the introduction of the Atmel Gallery and the Atmel Studio Extension Developer's Kit (XDK), it is now possible to release any development to the Atmel Gallery. Refer to the XDK User Guide.

5 Getting started with ASF and IAR Embedded Workbench

5.1 Install

1. Download IAR Embedded Workbench® on http://www.iar.com.
2. Install IAR.
3. Download the ASF standalone archive (.zip file) from http://www.atmel.com/asf.
4. Extract the ASF standalone archive on your hard drive (preferably on a location close to the root, to avoid any potential long name length issue on Windows®).

5.2 Header files update

For 8-bit AVR users it is recommended to update the toolchain header files, a description about how to do that is located in the readme.txt file under the xmega/utils/header_files/ directory.

For AVR UC3 users, it is recommended to update the IAR header files. To do that, unzip the avr32-headers.zip file (located under the avr32/utils/header_files/ directory) to the IAR EWAVR32 installation folder under "IAR installation folder"/Embedded Workbench x.y/avr32/inc/.

5.3 Navigating in the ASF standalone archive

The top folder organization is as follow:

- The avr32/ folder contains software modules (source code and projects) dedicated to AVR UC3 devices
- The mega/ folder contains software modules (source code and projects) dedicated to Atmel megaAVR devices
- The xmega/ folder contains software modules (source code and projects) dedicated to Atmel AVR XMEGA devices
- The common/ folder contains software modules (source code and projects) shared by all Atmel AVR devices
- The sam/ fodler contains software modules (source code and projects) dedicated to Atmel SAM devices
- The thirdparty/ folder contains software modules (source code and projects) from thirdparty providers for all AVR devices

The thirdparty/ folder is organized by thirdparty software module (that is, one folder per thirdparty software module).

The avr32/, xmega/, mega/, sam/ and common/ folders are organized as follow:

- The drivers/ folder contains low-level software drivers for Atmel MCU on-chip resources (for example, cpu, usart, adc drivers)
- The boards/ folder contains board-specific source code files
- The utils/ folder contains files that are used by all other modules: it holds several linker script files, IAR & GCC pre-compiled libraries of some ASF modules, and C/C++ utility files with general usage defines, macros and functions
- The services/ folder contains application-oriented pieces of software that are not specific to boards nor chips (for example, FAT, TCP/IP stack, OS, JPEG decoder). For the common/ top folder, the services/ folder also contains chip-specific code

- The components/ folder offers, for each supported hardware component, a software interface to interact with the component (for example, memories like SDRAM, SD card, or display)
- The applications/ folder contains hefty examples of applications using services and drivers

5.4 Start ASF examples

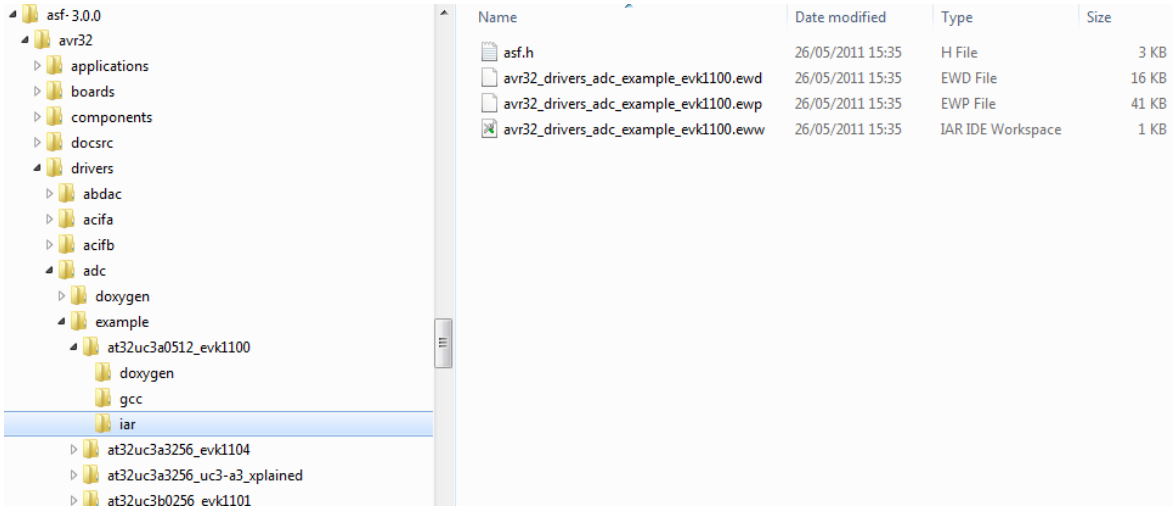
Using an example of usage of the Atmel AVR UC3 GPIO driver module, the IAR projects are located under:

- avr32/drivers/gpio/peripheral_bus_example/at32uc3a0512_evk1000/iar/ for an Atmel EVK1100 board
- avr32/drivers/gpio/peripheral_bus_example/at32uc3a0512_evk1105/iar/ for an Atmel EVK1105 board
- avr32/drivers/gpio/peripheral_bus_example/at32uc3b0256_evk1101/iar/ for an Atmel EVK1101 board
- avr32/drivers/gpio/peripheral_bus_example/at32uc3a3256_evk1104/iar/ for an Atmel EVK1104 board
- avr32/drivers/gpio/peripheral_bus_example/at32uc3l064_stk600-rcuc3l0/iar/ for an Atmel STK®600-RCUC3L0 setup
- avr32/drivers/gpio/peripheral_bus_example/at32uc3c0512c_uc3c_ek/iar/ for an Atmel AT32UC3C-EK board
- avr32/drivers/gpio/peripheral_bus_example/at32uc3l064_uc3l_ek/iar/ for an Atmel AT32UC3L-EK board

Each iar/ folder contains a full IAR project with:

- an IAR EWAVR32 workspace file (avr32_drivers_gpio_peripheral_bus_example_uc3l_ek.eww: double-click on this file to open the whole project)
- an IAR EWAVR32 project file (avr32_drivers_gpio_peripheral_bus_example_uc3l_ek.ewp)
- an IAR EWAVR32 debug configuration file (avr32_drivers_gpio_peripheral_bus_example_uc3l_ek.ewd)

Figure 5-1. IAR project location in the ASF.



5.5 Get ASF project documentation

5.5.1 Online Documentation

All ASF module and reference application Doxygen documentation can be found on http://asf.atmel.com.

5.5.2 Manual Documentation Generation (offline)

All modules are fully documented using doxygen tags. Each project within the Atmel ASF contains a doxyfile.doxygen (used to configure doxygen for a proper documentation generation): to generate the .html documentation, doxygen must be installed (see http://www.doxygen.org/download.html) and the doxyfile.doxygen must be used as the input configuration file for doxygen.

Using an example of usage of the Atmel AVR UC3 GPIO driver module as an example, for an Atmel AT32UC3C-EK board, the associated doxyfile.doxygen file is under the avr32/drivers/gpio/peripheral_bus_example/at32uc3c0512c_uc3c_ek/doxygen/ folder.

Run doxygen and use this doxyfile.doxygen as configuration file for doxygen.

Using the command line, this is done with the following command:

```
doxygen doxyfile.doxygen
```

The documentation entry file is:

```
avr32/drivers/gpio/peripheral_bus_example/at32uc3c0512c_uc3c_ek/doxygen/html/index.html
```

6 Getting started with ASF and GNU makefile

6.1 Install

1. Download the Atmel ASF standalone archive (zip file) from http://www.atmel.com/asf.
2. Extract the ASF standalone archive on your hard drive.
3. Install Atmel AVR Studio® 5 (http://www.atmel.com/avrstudio5) for the compiling and programming tools.
4. The software framework build system assumes that you have some basic software tools installed on the build machine. These tools are easy to install on most modern operating systems. On Microsoft® Windows systems these software applications are installed with the WinAVR package for 8-bit AVR (http://winavr.sourceforge.net/) and with the Atmel AVR GNU toolchain (http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4118) for 8-bit and 32-bit users, while on Linux® systems they are usually available using the distributions package system. Software which must be installed for the software framework to be useful:
   - make, sed, grep, sort, tac, bc, etc., often referred to as build essentials
   - sh – command interpreter (shell)

6.2 Header files update

For 8-bit AVR users it is recommended to update the toolchain header files, a description about how to do that is located in the readme.txt file under the xmega/utils/header_files/ directory.

For Atmel AVR UC3 users, it is recommended to update the IAR header files, a description about how to do that is located in the readme.txt file under the avr32/utils/header_files/ directory.

6.3 Navigating in the ASF standalone archive

The top folder organization is as follow:

- The avr32/ folder contains software modules (source code and projects) dedicated to AVR UC3 devices
- The mega/ folder contains software modules (source code and projects) dedicated to Atmel megaAVR devices
- The xmega/ folder contains software modules (source code and projects) dedicated to Atmel AVR XMEGA devices
- The common/ folder contains software modules (source code and projects) shared by all Atmel AVR devices
- The thirdparty/ folder contains software modules (source code and projects) from thirdparty providers for all AVR devices

The thirdparty/ folder is organized by thirdparty software module (that is, one folder per thirdparty software module).

The avr32/, xmega/ and common/ folders are organized as follow:

- The drivers/ folder contains low-level software drivers for AVR on-chip resources (for example, cpu, usart, adc drivers)
- The boards/ folder contains board-specific source code files

- The utils/ folder contains files that are used by all other modules: it holds several linker script files, IAR & GCC pre-compiled libraries of some Atmel ASF modules, and C/C++ utility files with general usage defines, macros and functions
- The services/ folder contain application-oriented pieces of software that are not specific to boards or chips (for example, FAT, TCP/IP stack, OS, JPEG decoder). For the common/ top folder, the services/ folder also contains chip-specific code
- The components/ folder offers, for each supported hardware component, a software interface to interact with the component (for example, memories like SDRAM, SD card, or display)
- The applications/ folder contains hefty examples of applications using services and drivers

6.4 Start ASF examples

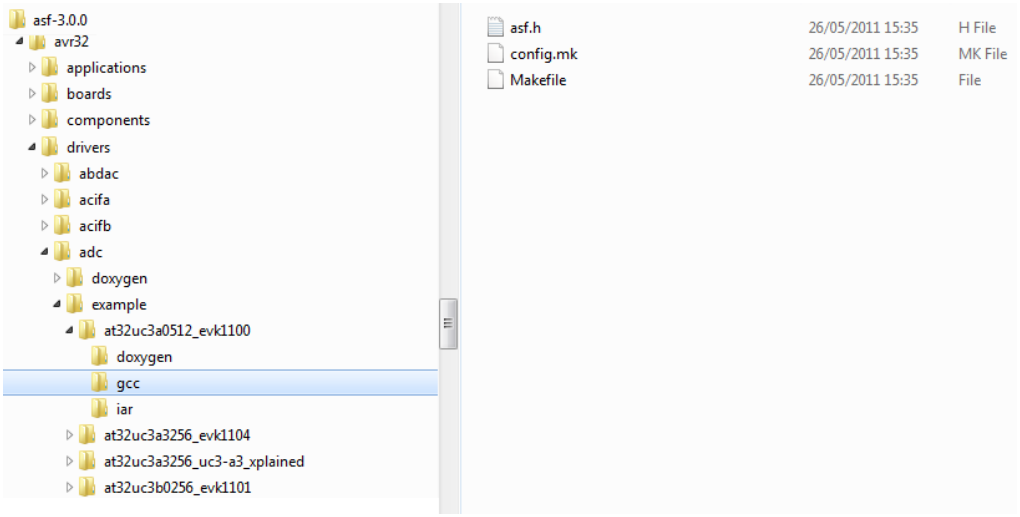
Using an example of usage of the Atmel AVR UC3 GPIO driver module as an example, the GCC projects are located under:

- avr32/drivers/gpio/peripheral_bus_example/at32uc3a0512_evk1000/gcc/ for an Atmel EVK1100 board
- avr32/drivers/gpio/peripheral_bus_example/at32uc3a0512_evk1105/gcc/ for an Atmel EVK1105 board
- avr32/drivers/gpio/peripheral_bus_example/at32uc3b0256_evk1101/gcc/ for an Atmel EVK1101 board
- avr32/drivers/gpio/peripheral_bus_example/at32uc3a3256_evk1104/gcc/ for an Atmel EVK1104 board
- avr32/drivers/gpio/peripheral_bus_example/at32uc3l064_stk600-rcuc3l0/gcc/ for an Atmel STK600-RCUC3L0 setup
- avr32/drivers/gpio/peripheral_bus_example/at32uc3c0512c_uc3c_ek/gcc/ for an Atmel AT32UC3C-EK board
- avr32/drivers/gpio/peripheral_bus_example/at32uc3l064_uc3l_ek/gcc/ for an Atmel AT32UC3L-EK board

Each gcc/ folder contains a GCC project with:

- a makefile
- a project configuration file config.mk

Figure 6-1. GNU makefile project location in the ASF.



6.5 Building the project

To build an application, simply enter the appropriate project directory (/gcc folder) and type make.

For example, for the Atmel AT32UC3A0512 GPIO peripheral bus example running on Atmel EVK1100 board, typing make in the avr32/drivers/gpio/peripheral_bus_example/at32uc3a0512_evk1100/gcc folder will result:

```
$ make
MKDIR   avr32/drivers/intc/
CC      avr32/drivers/intc/intc.o
MKDIR   avr32/drivers/gpio/
CC      avr32/drivers/gpio/gpio.o
MKDIR   avr32/drivers/gpio/peripheral_bus_example/
CC      avr32/drivers/gpio/peripheral_bus_example/gpio_periphe
MKDIR   avr32/utils/startup/
AS      avr32/utils/startup/startup_uc3.o
AS      avr32/utils/startup/trampoline_uc3.o
AS      avr32/drivers/intc/exception.o
LN      avr32_drivers_gpio_peripheral_bus_example_evk1100.elf
SIZE    avr32_drivers_gpio_peripheral_bus_example_evk1100.elf
avr32_drivers_gpio_peripheral_bus_example_evk1100.elf  :
section               size         addr
.reset              0x2004   0x80000000
.text                0x134   0x80002004
.exception           0x200   0x80002200
.rodata              0xa0    0x80002400
.dalign               0x4          0x4
.bss                 0xf0          0x8
.heap               0xef08         0xf8
.comment             0x30          0x0
.debug_aranges       0x158         0x0
.debug_pubnames      0x3ad         0x0
.debug_info         0x11ca         0x0
.debug_abbrev        0x554         0x0
.debug_line         0x1b6f         0x0
.debug_frame         0x294         0x0
```

```
.debug_str           0x8f0          0x0
.debug_loc           0x408          0x0
.debug_macinfo    0x2b7053          0x0
.stack              0x1000       0xf000
.debug_ranges        0x170          0x0
Total             0x2cdce5

   text    data     bss     dec     hex filename
 0x23d8     0x0  0xfffc   74708   123d4 avr32_drivers_gpio_per
e_evk1100.elf
OBJDUMP avr32_drivers_gpio_peripheral_bus_example_evk1100.lss
NM      avr32_drivers_gpio_peripheral_bus_example_evk1100.sym
OBJCOPY avr32_drivers_gpio_peripheral_bus_example_evk1100.hex
OBJCOPY avr32_drivers_gpio_peripheral_bus_example_evk1100.bin
```

The result of the build is located in the same folder as the makefile location:

- Elf file (.elf)
- Listing file (.lss)
- Symbol file (.sym)
- Hex file (.hex)
- Binary file (.bin)

6.6 Programming

6.6.1 Windows users

6.6.1.1 GUI-based solution

- Launch Atmel Studio
- Select “File->Open->Open Object File for Debugging” and select the generated .elf file. Select “Next”
- In the “Device Selection” menu, select the corresponding MCU. Click “Finish”
- The project is now available to program and debug. Refer to Atmel Studio getting started and user guide from the welcome screen

6.6.1.2 Command-line based solution for AVR Users

- Program utilities are installed with Atmel Studio
- Use atprogram.exe (e.g. in C:\Program Files (x86)\Atmel\Atmel Studio 6.0\avrdbg) as a programmer
- For example to program an at32uc3b0512 with a JTAGICE3 debugger:

```
atprogram -t jtagice3 -i jtag -d at32uc3b0512 program -f e:\file.elf
```

```
C:\Program Files (x86)\Atmel\Atmel Studio 6.0\avrdbg>atprogram.exe
AVR Studio Command Line Interface
Copyright (C) 2011 Atmel Corporation.

Usage: atprogram [options] <command> [arguments] [<command> [arguments] ...]
Options:
  -t  --tool <arg>           Tool name: avrdragon, avrispmk2, avrone, jtagice3,
                             jtagicemkii, qt600, stk500, stk600 or samice.
  -s  --serialnumber <arg>   The programmer/debugger serialnumber. Must be
```

```
                              specified when more than one debugger is
                              connected.
  -c  --comport <arg>         The com port to use for the programmer/debugger.
                              e.g. -c COM1 or -c 1
  -i  --interface <arg>       Physical interface: aWire, debugWIRE, HVPP, HVSP,
                              ISP, JTAG, PDI or TPI.
  -d  --device <arg>          Device name. E.g. atxmega128a1 or at32uc3a0256.
  -v  --verbose               Verbose output (debug).
  -h  --host <arg>            Target host which runs the avrdbg process.
  -p  --port <arg>            Specify which port to use for the avrdbg process.
                              Ignored if the host option is not given.
  -f  --force                 Force command even if firmware is not up to date.
  -cl --clock <arg>           The frequency to use for communication with a
                              device (hz, khz, mhz, default hz). E.g. -cl 10mhz
  -mb --max-baudrate <arg>    The maximum baud rate for communication on awire
                              (Kbps).
  -xr --externalreset         Apply external reset when starting a session.
  -dc --daisychain <args,..> Set up a JTAG daisychain. Arguments are <devices-
                              before devices-after instr-before instr-after>.
  -tv --target-voltage <arg> Set the STK600 or STK500 target voltage (float
                              value).
  -a0 --aref0 <arg>           Set the STK600 Aref0 or STK500 aref generator
                              voltage (float value).
  -a1 --aref1 <arg>           Set the STK600 Aref1 generator voltage (float
                              value).
  -cg --clock-generator <arg> Set the STK600 or STK500 Clock generator frequenc

                              (hz, khz, mhz, default hz).
      --timeout <arg>         Set the timeout value in seconds for commands. The
                              default is 180 seconds. Set to 0 for no timeout.
  -q  --quiet                 Do not display activity indicator.
  -?  --help                  Display help information.

Commands:
   chiperase  Full erase of chip.
   erase      Erase the specified memory.
   help       Displays help for a specific command.
   info       Display information about a device.
   list       Detect and print information about connected Atmel Tools.
   program    Program device with data from <file>.
   read       Read the contents of the memory on the device.
   secure     Set the security bit on UC3 and ARM devices.
   verify     Verify content of memory based on a file.
   version    Display the version.
   write      Write to the memory with values entered on the command line.
```

6.6.1.3 Command-line based solution for SAM Users

Refer to Atmel SAM-BA® In system programmer:

http://www.atmel.com/tools/ATMELSAM-BAIN-SYSTEMPROGRAMMER.aspx

6.6.2 Linux users

- For 8-bit users, use AVRDUDE for MCU programming. (http://www.nongnu.org/avrdude/)
- For 32-bit AVR users, refer to “AVR32 Studio Users” section for a GUI-based solution
- For ARM users, refer to SAM-BA In System programmer (http://www.atmel.com/tools/ATMELSAM-BAIN-SYSTEMPROGRAMMER.aspx)

6.7 AVR32 Studio users

32-bit AVR users can use Atmel ASF GNU makefile in 32-bit Atmel AVR Studio. It is possible to work with an unzipped ASF package from within 32-bit AVR Studio: this is described in the application note "AVR32769: How to Compile the standalone AVR32 Software Framework in AVR32 Studio V2": http://www.atmel.com/dyn/resources/prod_documents/doc32115.pdf.

6.8 Access to project documentation

6.8.1 Online Documentation

All ASF module and reference application Doxygen documentation can be found on http://asf.atmel.com.

6.8.2 Offline Documentation

All modules are fully documented using doxygen tags. Each project within the ASF contains a doxyfile.doxygen (used to configure doxygen for a proper documentation generation): to generate the html documentation, doxygen must be installed (see http://www.doxygen.org/download.html) and the doxyfile.doxygen must be used as the input configuration file for doxygen.

Using an example of usage of the Atmel AVR UC3 GPIO driver module as an example, for an Atmel AT32UC3C-EK board, the associated doxyfile.doxygen file is under the avr32/drivers/gpio/peripheral_bus_example/at32uc3c0512c_uc3c_ek/doxygen/ folder.

Run doxygen and use this doxyfile.doxygen as configuration file for doxygen.

Using the command line, this is done with the following command:

```
doxygen doxyfile.doxygen
```

The documentation entry file is:

```
avr32/drivers/gpio/peripheral_bus_example/at32uc3c0512c_uc3c_ek/doxygen/html/index.html
```

7 Table of contents

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg.
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370